# Hockey Pool Linear Regression

Derek Knight

2023-12-12

## Data Preparation and Library Import

In this section, we begin by setting up our environment and preparing the data for analysis. The steps involved are as follows:

Installing and Loading Necessary Libraries

We start by installing and then loading the car package. This package provides functions for regression diagnostics and other statistical tools which are essential for our analysis.

## Loading the Dataset

Next, we load our datasets from CSV files. Two datasets are used in this analysis:

```
QuantHockey Dataset - Less Features LR.csv: This dataset contains selected features and is used for the
QuantHockey Dataset Test.csv: This dataset is used for our predications.
```

```r
class(df)
```

```
## [1] "function"
```

```r
df <- read.csv("QuantHockey Dataset - Less Features LR.csv")
test_df <- read.csv("QuantHockey Dataset Test.csv")
```

## Regression Model Building

The steps include:

Renaming columns, defining the response variable, and fitting the model.

- Selection of Regressors: A subset of variables is chosen as regressors based on their potential influence on the response variable.
- Fitting the Linear Regression Model: Fit the linear regression model to our data.
- Model Summary and ANOVA: The summary of the regression model provides insights into the fit and ANOVA is performed to compare models or test certain model assumptions.

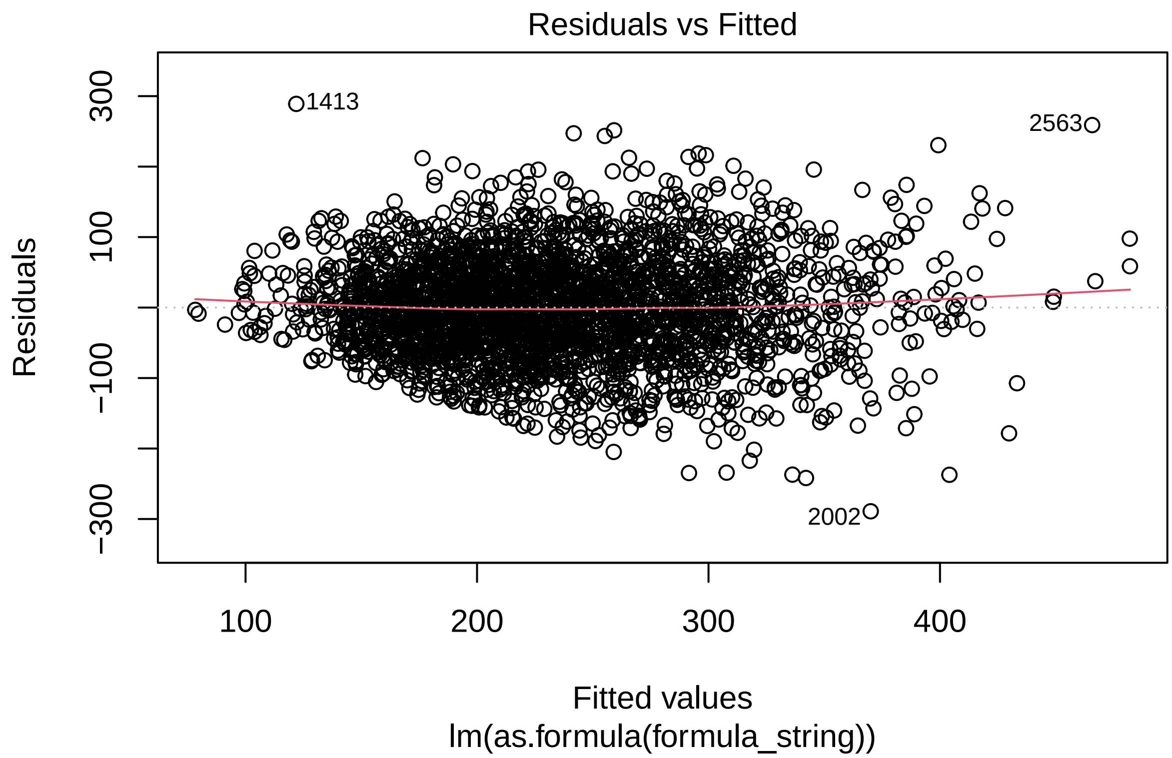
```r
colnames(df) <- c("Rk", "Age", "GP", "G", "A", "P", "PlusMinus", "PPG", "SHG", "GWG", "OTG",
                  "PPA", "SHA", "GWA", "PPP", "SHP", "GWP", "OTP", "PPPPercent", "GoalsPer60",
                  "AssistsPer60", "PointsPer60", "ESGoalsPer60", "ESAssistsPer60", "ESPointsPer60",
                  "PPGoalsPer60", "PPAssistsPer60", "PPPointsPer60", "GoalsPerGame", "AssistsPerGame",
                  "PointsPerGame", "Shots", "ShootingPercent", "Hits", "BS", "Year", "FantasyPoints",
                  "NextYearFantasyPoints")

# Define the response variable
y <- df$NextYearFantasyPoints

# Fit a linear regression model with selected regressors
selected_regressors <- c("Rk", "Age", "GP","G", "A", "PPG",
                         "PPA", "SHA", "GWG",
                         "Shots", "Hits", "BS")

# Include only the selected regressors in the formula
formula_string <- paste("y ~", paste(selected_regressors, collapse = " + "))

# Fit a linear regression model with selected regressors
model <- lm(as.formula(formula_string), data = df)

# View regression summary
summary(model)
```

```
##
## Call:
## lm(formula = as.formula(formula_string), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -289.07  -47.72   -2.18   46.26  288.99
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 335.09451   13.56131  24.710  < 2e-16 ***
## Rk           -0.17258    0.03256  -5.301 1.24e-07 ***
## Age          -3.06351    0.32213  -9.510  < 2e-16 ***
## GP           -2.46297    0.14872 -16.561  < 2e-16 ***
## G             0.88713    0.37929   2.339 0.019408 *
## A             0.95110    0.27232   3.493 0.000486 ***
## PPG           1.55487    0.61378   2.533 0.011354 *
## PPA           1.31819    0.43467   3.033 0.002446 **
## SHA           5.00185    1.93301   2.588 0.009714 **
## GWG           1.50914    0.90766   1.663 0.096488 .
## Shots         0.60340    0.04363  13.830  < 2e-16 ***
## Hits          0.20134    0.03032   6.639 3.76e-11 ***
## BS            0.14194    0.04666   3.042 0.002370 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.56 on 2834 degrees of freedom
## Multiple R-squared:  0.4175, Adjusted R-squared:  0.4151
## F-statistic: 169.3 on 12 and 2834 DF,  p-value: < 2.2e-16
```

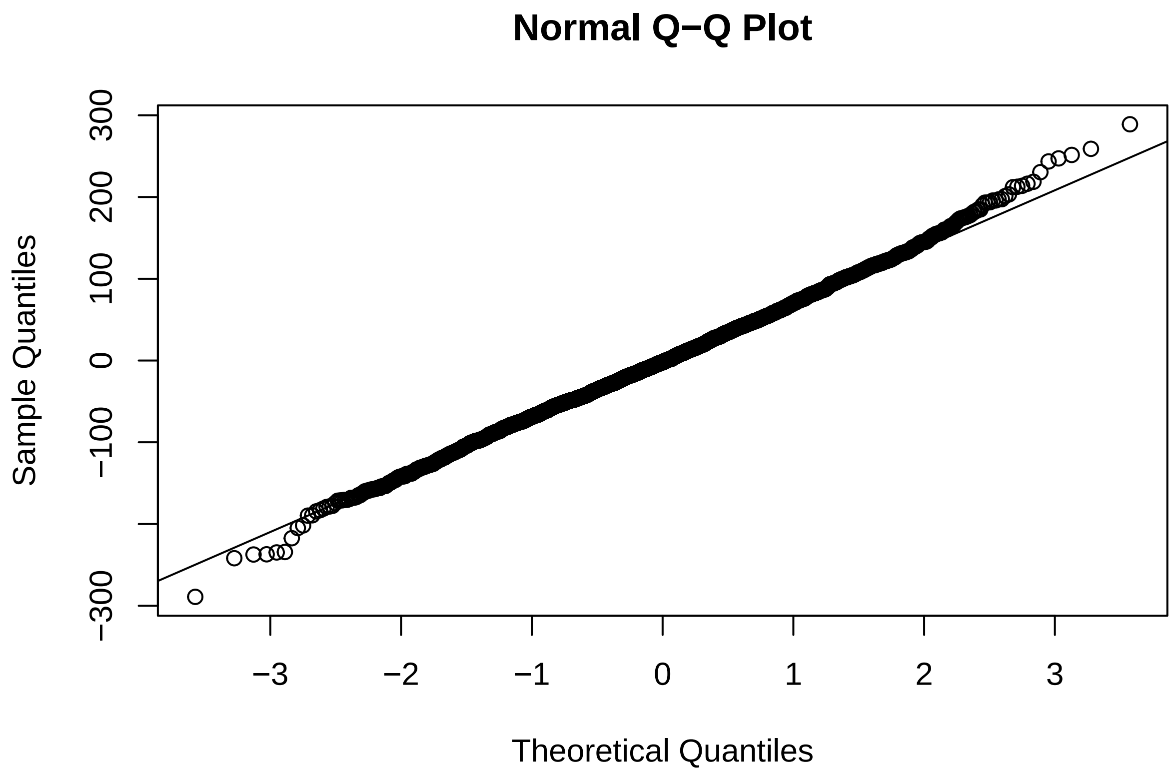
```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##             Df   Sum Sq Mean Sq   F value    Pr(>F)
## Rk           1  7023910 7023910 1371.7706 < 2.2e-16 ***
## Age          1   488148  488148   95.3354 < 2.2e-16 ***
## GP           1    51579   51579   10.0735  0.001520 **
## G            1   434727  434727   84.9022 < 2.2e-16 ***
## A            1   490117  490117   95.7200 < 2.2e-16 ***
## PPG          1   152181  152181   29.7210 5.418e-08 ***
## PPA          1   123088  123088   24.0392 9.972e-07 ***
## SHA          1    69448   69448   13.5632  0.000235 ***
## GWG          1    20664   20664    4.0356  0.044644 *
## Shots        1  1232403 1232403  240.6884 < 2.2e-16 ***
## Hits         1   268524  268524   52.4427 5.681e-13 ***
## BS           1    47389   47389    9.2550  0.002370 **
## Residuals 2834 14510999    5120
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

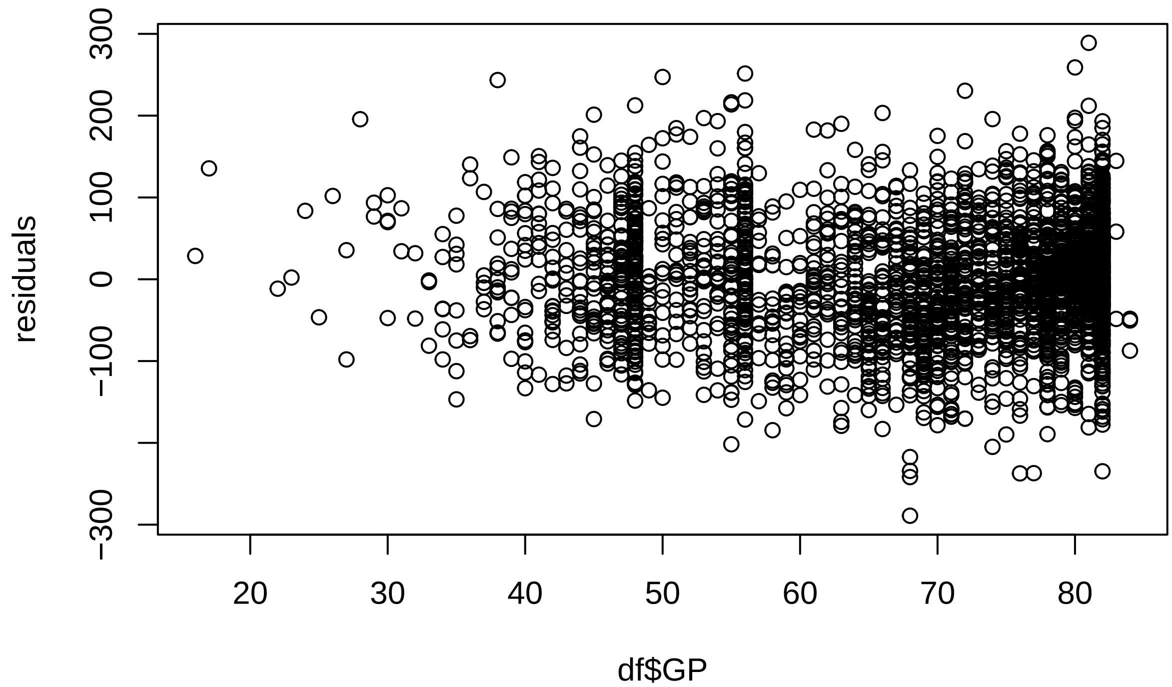
```r
residuals <- resid(model)
# Residual Plot:
# Residual plot to visually inspect the distribution of residuals and identify
plot(model, which = 1)
```



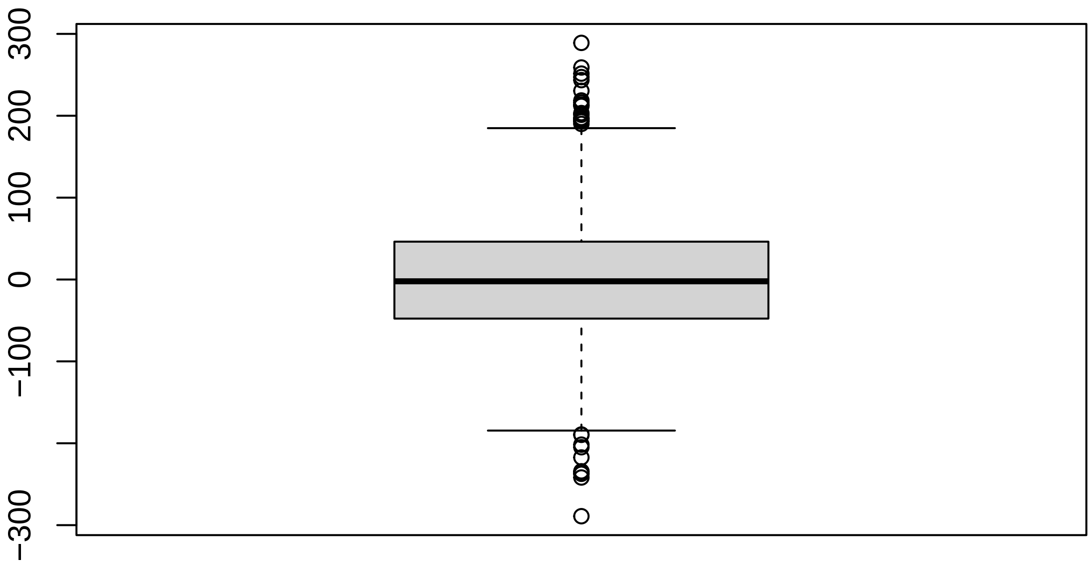

```
# Normality Check:
# Check for the normality of residuals using a Q-Q plot:
qqnorm(residuals)
qqline(residuals)
```



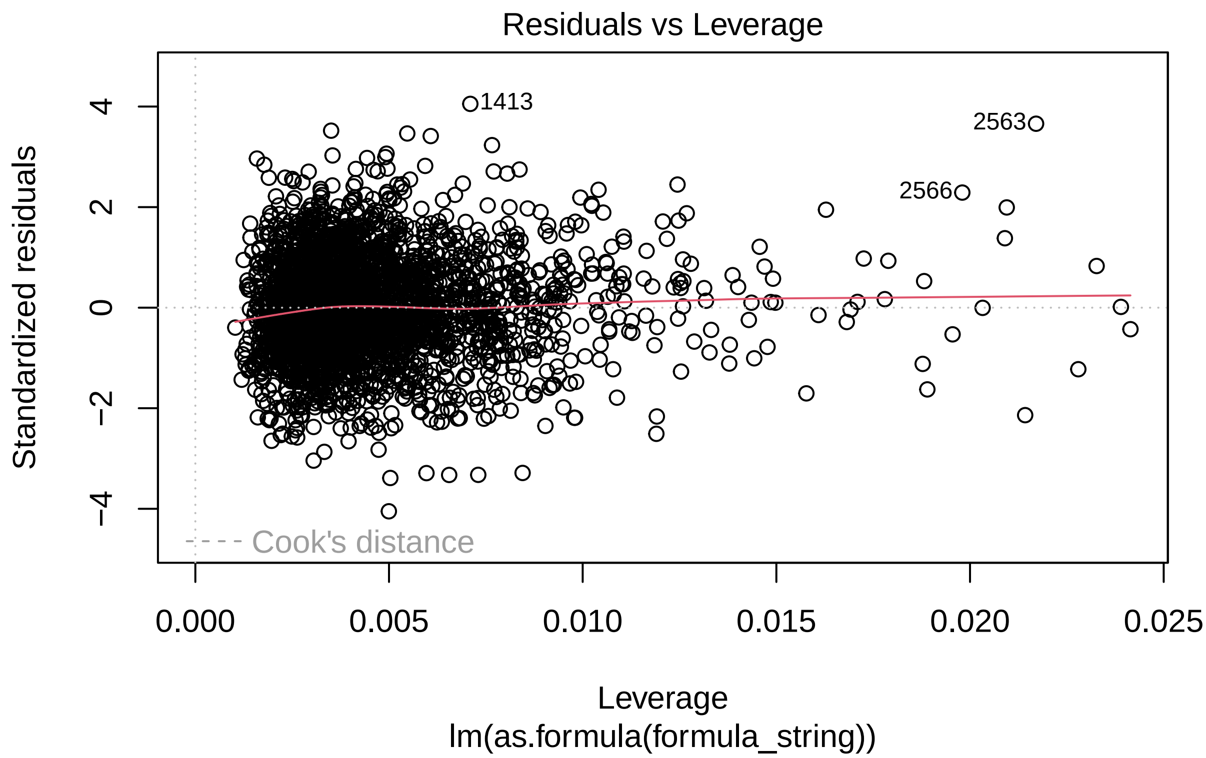

```r
# Homoscedasticity Check:
# Check for constant variance (homoscedasticity) by plotting residuals against the predictor variable
plot(df$GP, residuals)
```




```r
# Check for Outliers:
# Identify potential outliers by examining the residuals.
boxplot(residuals)
```




```r
plot(model, which = 5)  # Leverage plot
```




```
# Influence Points:
# Identify influential observations using Cook's distance:
infl <- influence.measures(model)
# plot(infl, which = 4)  # Cook's distance plot
```

## Generating Predictions and Exporting Results

### Making Predictions

Using the predict() function, we generate predictions from our linear regression model for the test_df dataset, which contains data we want to evaluate but wasn't used in the model training phase.

### Combining Predictions with Original Data

To facilitate easy analysis and comparison, we combine the predictions with the original test data. This merged dataset includes all the features from test_df along with a new column Predictions that holds our model's predictions.

### Exporting the Results

Finally, we export the combined dataset with predictions to a CSV file. This file, named predictions.csv, can be used for detailed analysis, visualization, or presentation of the model's output.

```r
predictions <- predict(model, newdata = test_df)

# Combine the original data with the predictions
result_data <- cbind(test_df, Predictions = predictions)

# Specify the file path for the CSV file
csv_file_path <- "predictions.csv"

# Export the combined data to a CSV file
write.csv(result_data, file = csv_file_path, row.names = FALSE)
```